⑲日本国特許庁（JP）　⑪特許出願公開

⑫公開特許公報（A）　平4－137046

⑤Int.Cl.⁵	識別記号	庁内整理番号
G 06 F 9/46	340 F	8120－5B
9/06	410 A	7927－5B
11/32	A	7165－5B

⑬公開　平成4年(1992)5月12日

審査請求　未請求　請求項の数 2　（全11頁）

⑭発明の名称　電子計算機のオペレーティングシステム

⑳特　願　平2－259003
㉒出　願　平2(1990)9月28日

㉑発明者　伊藤　聡　神奈川県川崎市幸区小向東芝町1番地　株式会社東芝総合研究所内
㉑出願人　株式会社東芝　神奈川県川崎市幸区堀川町72番地
㉔代理人　弁理士　鈴江　武彦　外3名

明　細　書

1．発明の名称

電子計算機のオペレーティングシステム

2．特許請求の範囲

(1) 利用者が用意したプログラムをコンパイルし、必要なモジュールをリンクし、作成されたロードモジュールを実行する電子計算機のオペレーティングシステムにおいて、

利用者が用意したプログラムのどこを実行しているかを監視する手段と、前記電子計算機の計算機資源の使用量を監視する手段と、前記各手段による監視結果に基づき、使用している計算機資源が指定された上限値に所定量まで近付いたか又は該上限値を超えたかを判定する手段と、該手段により計算機資源が指定された上限値に所定量まで近付いたか又は該上限値を超えたと判定された場合に、ロードモジュール，全レジスタ及びステータスワードレジスタの内容をダンプして処理を一時的に終了させ、且つ該プログラムを実行するには計算機資源が足りなかった旨を利用者に通報する手段と、利用者が定められた手順を踏んだ場合に、前記ダンプした内容を元に戻して前記中断点からの再実行を行う手段とを具備してなることを特徴とする電子計算機のオペレーティングシステム。

(2) 利用者が用意したプログラムをコンパイルし、必要なモジュールをリンクし、作成されたロードモジュールを実行する電子計算機のオペレーティングシステムにおいて、

利用者が用意したプログラムのどこを実行しているかを監視する手段と、前記電子計算機の計算機資源の使用量を監視する手段と、前記各手段による監視結果に基づき、使用している計算機資源が指定された上限値に所定量まで近付いたか又は該上限値を超えたかを判定する手段と、該手段により計算機資源が指定された上限値に所定量まで近付いたか又は該上限値を超えたと判定された場合に、ロードモジュール，全レジスタ及びステータスワードレジスタの内容を外部記憶媒体にダンプして処理を一時的に終

了させ、且つ該プログラムを実行するには計算機資源が足りなかった旨を利用者に通報する手段と、利用者が定められた手順を踏んだ場合に、前記ダンプした内容を元に戻して前記中断点からの再実行を行う手段とを具備してなり、

前記外部記憶媒体は少なくとも２種類設置され、前記ロードモジュール，全レジスタ及びステータスワードレジスタの内容をダンプする際に、最もアクセス時間の短い記憶媒体にフルダンプを行い、システムが決めた所定の時間を過ぎても予め決められた手順に従った指示がなかったときは、次にアクセス時間の短い記憶媒体に該ダンプ内容をマイグレートさせ、次々にアクセス時間のより長い記憶媒体に該ダンプ内容をマイグレートさせることを特徴とする電子計算機のオペレーティングシステム。

3．発明の詳細な説明

［発明の目的］

（産業上の利用分野）

本発明は、利用者がそれぞれの問題に応じたプログラムを用意し、これに従って処理を行う所謂ノイマン型（プログラム内蔵型）の電子計算機のオペレーティングシステムに係わり、特に計算機システムを効率良く使用するための電子計算機のオペレーティングシステムに関する。

（従来の技術）

通常の電子計算機、特にメインフレーム型の計算機のソフトウエアは、利用者が処理手順を指定するユーザ定義プログラムとシステム側のオペレーティングシステム（ＯＳ）から構成されている。ユーザ定義プログラムは、個々の問題に応じた計算の手順を計算機側に指示するものである。計算機によってこのプログラムの言語仕様が異なっていては利用し難いので、現在では標準化が進んでおり、FORTRAN，BASIC，ALGOL，C，PASCAL，COBOL などの高級言語が一般的に利用されている。これに対して、オペレーティングシステムは計算機固有のソフトウエア群であり、該高級言語で書かれた利用者定義プログラムのコンパイル，リンク，実行を行う処理系と、通常の計算機ではマルチプロセス環境を提供しなければならないので、個々のジョブの管理やスケジューリング，計算機資源の管理と監視，ユーザ利用環境の提供などを行う制御系とがあり、実際には計算機を動かせるのに必要な全てがこれで行われる。

計算機のオペレーティングシステムは、現時点では基本的には標準化されておらず計算機毎に異なるが、近年ではこの標準化の機運が高まり、ＵＮＩＸがその実質的な標準と見なされつつある。また、小型の計算機ではＭＳ－ＤＯＳがその地位にある。ごく最近では、ＵＮＩＸをさらに発展させたタイプのＭＡＣＨやＭＳ－ＤＯＳを発展させたＯＳ／２なども提案され、実装している計算機も現れている。

オペレーティングシステムが行うべき重要な仕事に、計算機資源の管理と監視がある。計算機資源とは、そのジョブが使用できるハードウエア（主記憶領域，外部記憶装置としての半導体記憶装置，磁気記憶装置など）とソフトウエア（高級言語で書かれたプログラムを処理するためのコンパイラ，リンカ，ローダ，システム側の提供するユーティリティ，汎用のソフトウエアなど）とを通常指す。

計算機資源をどのくらい使うかは、実行しようとしているユーザ指定のプログラムに依存する。ユーザ指定のプログラムが要求する資源を計算機側が可能な範囲で無制限に与える方式もあるが、特に汎用大型計算機のように多数の利用者のジョブをマルチタスク方式で処理するものはシステム全体の効率を考えるため、ユーザ指定のプログラムに無関係に与える計算機資源はデフォルト値が決められている。これによって、特にユーザ定義プログラムの誤りにより生じる無駄な計算や論理的・実質的に終了しない無限回の演算ループを避けることができる。そうはいっても、これでは実行できないジョブもある訳で、これに対しては使用できる計算機資源の上限値を利用者が変更しなければならない。

通常の汎用大型計算機では、ジョブ制御言語

仕様の中にそのジョブで使用できる計算機資源の制限が指定できるようになっている。例えば、そのジョブで使用する主記憶装置の容量の上限値やＣＰＵ時間の上限値などである。しかし、処理に必要な計算機資源の正確な見積もりは極めて難しく、後少しで結果が出るのに計算機資源が無くなってジョブが打ち切られてしまうことが、実際の利用ではよく起こる。小さいジョブであれば、利用できる計算機資源を大きくして再実行すればよいが、大きなジョブの場合はこれは大変な無駄になる。従来の計算機では、これに対する考慮はなにもされていなかった。

通常、大規模なプログラムの開発では、テスト用のデータというものがあり、このデータとして直ぐ計算が終わり、その結果の妥当性が簡単にチェックできるものを用意して、プログラムが正しく働いているかどうかを調べる。従って、プログラムが要求する主記憶容量や磁気記憶容量などはこうした試行計算で判明する。一方、計算時間の方はテストケースは直ぐに終わる場合を想定して行うので、これは余り参考にならない。科学技術計算では、よく反復計算を行って要求される精度まで計算値が収束したら実行を終了させることを行うが、このような場合、先験的に計算時間（ＣＰＵ時間）を見積ることは不可能に近い。特に、大規模な計算の場合には計算時間（ＣＰＵ時間）の僅かな過少見積りが、大きな無駄に結び付く。

計算機資源の見積りの難しさという点では、上記の計算時間（ＣＰＵ時間）だけではなく、磁気記憶装置の容量についてもいえる。現在の大半の大型計算機では恒久的にデータを保存するには外部磁気記憶装置上にデータファイルを作成して、これを利用する。利用者が磁気記憶装置を使用する際には、まずその容量を指定しなければならない。磁気記憶装置の容量の大きさは大体データ量（語数，一語の長さなどから見積もることが可能である）に合わせて要求するが、特に利用者が初心者である場合は、見積りに失敗することが多い。そして、実質的な計算は完全に終了しているにも拘らず、計算が終了して最後に磁気記憶装置にデータを掃き出している時点で処理しきれない量を書き込もうとして、処理が異常終了することも起きる。

最近の計算機ではこれを回避するために、一次割当て量と増分量というものが指定できるようになっている。一次割当て量は初期に要求する量であり、増分量とは一次割当て量を使い切った際に増分する量であり、この増分量に従って規定された回数（例えば１０回）恒久的なファイルの容量を拡張する。このような方法であっても、計算機資源を使い切って失敗することが起こり得る。

計算機資源を使い切るということは一人のユーザから見ると良く起きることであるが、一方、これは計算機システムから見れば本当に資源を使い切っているわけではない。ＣＰＵ時間は計算機側から見れば無尽蔵にあるし、記憶容量にしても計算機側から見ればまだまだ余裕があるのが普通である。しかし、計算機側はマルチタスクで動いていることを重視して、通常は制限を設けているのである。即ち、一つのタスクにあまりにも大きな計算機資源を割り当てると他のタスクが動かなくなることを避けている。しかし、ジョブが異常終了すれば、これを変更して再実行させるのであるから、無駄な計算を行わせることになる。このような再計算に伴う無駄も考慮した場合、こうした制限が計算機の全体としての効率的な利用、全体としての（必要な結果を得るという意味で）スループットを向上させる、という観点からどれだけ妥当なことなのか、疑問の余地はある。

ジョブの再始動及び据え置き再始動機能は最近の大型計算機にも用意されている場合がある。しかし、これらは基本的にジョブが異常終了した場合の後始末をどう行うかを規定しているに過ぎず、さらに大抵は計算機の操作員（オペレータ）が操作できるのみである場合が多く、計算機資源を使い果たした場合に限った取扱いはなされていない。予想以上に計算機資源を使っ

てしまうのは、もちろん広くいえば利用者側の使い方の誤りであるが、単にＣＰＵ時間や記憶容量の見積もり違い程度であることが多い。従って、これらの「罪の軽い誤り」に対しては通常の異常終了と別けて考えることが、利用者側にとってもまた先程述べたように計算機側にとっても重要である。

（発明が解決しようとする課題）

このように、従来の電子計算機においては、利用者が予め設定した計算機資源がなくなると、ジョブが打ち切られてしまい、途中まで処理した結果が無駄になる。予め設定する計算機資源を十分に大きくしておけば、このような問題は発生しないが、この場合は有限の計算機資源を無駄に使うことにつながる。つまり、従来方式では計算機の利用効率が低下するという問題があった。

本発明は、上記事情を考慮してなされてもので、その目的とするところは、予め設定した計算機資源を使い切った後でも、その中断点からの再実行を可能とすることができ、計算機の利用効率の向上をはかり得る電子計算機のオペレーティングシステムを提供することにある。

［発明の構成］

（課題を解決するための手段）

本発明の骨子は、計算機資源を使い切ったときにそのジョブを異常終了させて実行を打ち切るのではなく、その時点で、利用者側の判断を仰ぐことを取り入れたオペレーティングシステムを構築することにより、再計算などの無駄を省くことある。

即ち本発明は、利用者が用意したプログラムをコンパイルし、必要なモジュールをリンクし、作成されたロードモジュールを実行する電子計算機のオペレーティングシステムにおいて、利用者が用意したプログラムのどこを実行しているかを監視する手段と、電子計算機の計算機資源の使用量を監視する手段と、これらの各手段による監視結果に基づき、使用している計算機資源が指定された上限値に所定量まで近付いたか又は該上限値を超えたかを判定する手段と、該手段により計算機資源が指定された上限値に所定量まで近付いたか又は該上限値を超えたと判定された場合に、ロードモジュール，全レジスタ及びステータスワードレジスタの内容を外部記憶媒体にダンプして処理を一時的に終了させ、且つ該プログラムを実行するには計算機資源が足りなかった旨を利用者に通報する手段と、利用者が定められた手順を踏んだ場合に、ダンプした内容を元に戻して中断点からの再実行を行う手段とを具備し、さらに望ましくは、予め定められた期間中に利用者或いは操作員が所定の手続きを取らなかったときには、ダンプした内容を消去する手段を備えたことを特徴としている。

また本発明は、上記の構成に加え、外部記憶媒体を少なくとも２種類設置し、ロードモジュール，全レジスタ及びステータスワードレジスタの内容をダンプする際に、最もアクセス時間の短い記憶媒体にまずフルダンプを行い、システムが決めた所定の時間を過ぎても予め決められた手順に従った指示がなかったときは、次にアクセス時間の短い記憶媒体に該ダンプ内容をマイグレートさせ、さらにこの記憶媒体上にあるうちにも指示がないときは、さらにアクセス時間の掛かる記憶媒体に該ダンプ内容をマイグレートさせていくといった、次々にアクセス時間のより長い記憶媒体に該ダンプ内容をマイグレートさせることを特徴としている。

（作用）

本発明によれば、使用している計算機資源が上限値に所定量まで近付いたか又は上限値を越えた場合、処理が一時的に終了されると共に、コードモジュール，全レジスタ及びステータスワードレジスタの内容が外部記憶装置にダンプされる。従って、利用者が予め定められた手順を踏めば、ダンプした内容を元に戻して中断点からの再実行を行うことが可能となる。そしてこの場合、最初から処理をやり直すのと違い、中断した時点までの処理結果を利用できるので、

再計算の無駄をなくして計算機の利用効率の向上をはかり得る。しかも、予め設定する計算機資源量を十分に大きくする必要もないことから、計算機資源を有効に使うことができ、これによっても計算機の利用効率向上をはかることが可能である。また、次々にアクセス時間のより長い記憶媒体に該ダンプ内容をマイグレートさせ、さらにある決められた期間内に再起動の命令が出されないときは、ダンプした内容を消去することにより、記憶媒体を有効に使うことが可能となる。

（実施例）

以下、本発明の詳細を図示の実施例によって説明する。

第１図は本発明の一実施例に係わるオペレーティングシステムを使用した電子計算機のシステム構成例を示す図である。図中１０は計算機本体で、この計算機本体１０は中央処理演算部（ＣＰＵ）１１，入出力処理装置１２，利用者が用意したプログラムのどこを実行しているかを監視するプログラム監視部１３，割り込み制御部１４，計算機資源の使用量を監視する計算機資源監視部１５，半導体メモリからなる主記憶装置１６，汎用及び演算用レジスタ群１７などから構成されている。

また、２１は端末制御装置、２２は複数の端末、２３はカードリーダ、２４はプリンタであり、３０は半導体メモリ３１，磁気ディスク３２，磁気テープ３３及び光ディスク３４からなる外部記憶装置である。また、４０は半導体メモリ４１，磁気ディスク４２及び磁気テープ４３からなる外部記憶装置であり、この記憶装置４０はオートマチックファイルマイグレーションコントローラ（ＡＦＭＣ）５０により制御されている。ＡＦＭＣ５０は、計算機本体１０からダンプした内容を、半導体メモリ４１→磁気ディスク４２→磁気テープ４３の順にマイグレーションするものである。

本発明の実施例では、主に計算機資源としてＣＰＵ時間をとって説明するが、それ以外の計算機資源でも同様の実施例を構築することができる。また、多くの利用者にとってもっとも枯渇しやすい計算機資源はＣＰＵ時間であるのでこれを例として、構成例の詳細について述べる。

まず、計算機資源を観測するので、ＣＰＵ時間を随時監視する手段が必要であるが、通常の計算機は必ずタイマを備えており、ジョブ管理の面から、各ジョブのそれぞれのステップで消費している時間を監視している。従って、本発明のために新たな監視手段（ハードウェア）を付ける必要はない。さて、ユーザのプログラムを処理しながら、ＣＰＵ時間を計測し、これが設定された値に近付いたとする。例えばジョブが使用できる時間の９５％を使い切ったとしよう。このとき、この中断点からの再実行に備えて、実行時のフルダンプをとる。これには、使用している変数に対応するレジスタの内容とそのアドレス及びプログラムステータスワード（ＰＳＷ）の内容と番地、ポインタの位置、及びロードモジュールの全てを保存すればよい。

現在の計算機では、通常マルチタスク方式でジョブを処理しており、タスクの切り替えは実は常に行われていることである。タスク切替えの大まかな手順を、第２図に示す。タスクＡを実行中にＯＳへの割り込みがあると、割り込み処理ルーチンに移り、タスクＢを選択する。そして、再起動の準備を行ったのち、タスクＢを実行する。また、タスクＢの実行中にＯＳへの割り込みがあると、上記と同様に割り込み処理ルーチンに移り、タスクＡを選択し、再起動準備を行ったのち、タスクＡを実行する。

演算処理は計算機内では最も速く処理されるのに対し、記憶領域へのアクセスには時間が掛かる。また、種々の監視用のタスクやシステムチェック用のタスクなどもあって、ＣＰＵをこれらが使用する場合には必ずタスクの切替えが必要である。タスクの切替えが起こる原因は、イベントドリブンによるものと時分割によるものとがある。前者の場合は直接割込み命令（スーパバイザーコール割込み・ＳＶＣなど）が入

ってくるのに対して、後者では一度ＯＳに要求を出す。どちらの場合も割込み処理ルーチンに処理は移り、ＣＰＵの全レジスタ、ＰＳＷレジスタの内容や番地が保存される。次に、別のタスクがある基準によって選択され、これに対応するＧＲ、ＰＳＷの内容がロードされて、処理が再開される。

計算機の演算処理装置（ＣＰＵ）内の手続きには、一つのジョブに対して上記のような流れがある。ハードウェアは勿論これに対して作られているわけであるが、タスク切替えを行って別のタスクに対しても同一のハードウェアで処理を行う。通常のタスク切替えの場合はこの保存された内容を主記憶上に常駐させることが多い。これは、タスクの切替えを高速で行うためである。しかし、主記憶容量に余裕のない時は、外部乃至補助記憶装置に一時的に掃き出す。このように主記憶領域からデータを一時的に追い出すことをスワップアウトすると言い、これに使用する領域をスワップ領域と言う。スワップ領域はハードウェア的には高速アクセス可能な記憶媒体であって、これも無限の大きさではない。

従って、スワップ領域の大小が同時に処理できるタスクの数の上限を決め、これより多いジョブはシスインキューで待ち行列に入ったまま、待たされる。ＣＰＵ時間を使い切ってダンプするときも、手順的にはこれと全く同じことである。但し、タスク切替えの場合は自動的に再実行行われるが、計算機資源を使い切った場合は、再実行は利用者或いはオペレータの指示に応じて行わせることになる。

通常、タスクを切り替える際、スワップインやスワップアウトに余計な時間を掛けないようにするために、ここに使用する記憶媒体はアクセス時間の特に短いもの、例えば半導体記憶素子（高速動作の出来るバイポーラ素子など）を使用する。しかし、この種の記憶素子は価格も高く、また消費電力も大きいので、このような記憶媒体を大量に実装することは技術的に困難であるし、経済的でもない。

さて、計算機のジョブ制御言語仕様で計算機資源を制限できるようにしてあるのは、大きな計算機資源を要求するジョブの処理の優先順位を下げて、全体としては効率良くジョブを処理するようにジョブをスケジューリングするためでもあるが、それよりも、ユーザプログラムの誤りによって無駄な演算を行ったり、特にプログラムマのミスによって無限ループから出てこれなくなるのを防ぐためである。このような状況に陥ったジョブについてもプログラムの中断時のフルダンプを高速動作可能な半導体記憶装置上に取るとなると、今度は記憶媒体の有効利用に反する。さらに、実行形式のプログラムは通常かなり大きなものであることが多く、特に科学技術計算では大規模な行列演算や数値積分などがあり、この場合のロードモジュールの規模は極めて大きい。例えば、1000×1000の行列要素を定義すれば、それだけで１メガワード必要である。

従って、これらを全てフルダンプしていたら、いかに記憶媒体の大きな計算機であっても記憶領域がすぐ足りなくなるし、保存しただけで再利用されないとしたらこれは全くの無駄になる。さらに、再実行のためにダンプするのであるから記憶装置のアクセス時間が若干かかっても問題ではない。しかし、再実行の際にこの読み出しに時間の掛かることは問題がないにしても、中断させるときにアクセス時間の掛かる記憶媒体であると、スワップアウトさせるのにＣＰＵがＩ／Ｏアクセス待ちばかり起こしてしまうので（記憶媒体がアクセス可能になるまで、ＣＰＵはなにもしないで待っているから）、ＣＰＵの無駄になる。

これを避けるには以下のように、アクセス時間の異なる複数個の外部記憶装置を用意する。ＣＰＵが直接スワップアウトさせる相手先はアクセス時間の掛からない高速記憶媒体（半導体記憶装置など）を使用する。このフルダンプが行われるのと同時にジョブそのものは終了させ

れられて、利用者にはフルダンプが行われたことがメッセージなどで知らされる。

さて、ダンプされたファイルはある一定時間の内に再起動が要求されないとき、よりアクセス時間の掛かる記憶媒体（例えば磁気ディスク記憶装置など）にマイグレートされる。高速記憶媒体からより遅い記憶媒体へのマイグレートはＣＰＵの介在なしに行うことが可能であるので、ここでアクセス速度が遅くてもこれによってＣＰＵがＩ／Ｏリクエストを出して待ってしまうことは起きない。さらに、ここにダンプファイルが一定の時間内に利用者から再起動乃至消去の指示なければ、さらにアクセス時間の掛かる記憶媒体（磁気テープ式記憶装置など）にさらにマイグレートすることができる。

ここで例として述べた記憶装置は、高速アクセス可能なものから順に言って、半導体メモリ，磁気ディスク，磁気テープの順であった。これは、この順でビット当りの価格の掛かるものでもあるし、また消費電力の大きい順でもある。また、記憶容量という観点からは小規模なものから大規模なものになっている。磁気テープの場合は、自動的にマウント・アンマウントする大規模記憶装置（マスストレージシステム・ＭＳＳ）も実用化されているので、これを使用すれば記憶容量は実際には無尽蔵にあると考えてもよい。

ここで、計算機資源を使い切ったのは利用者の誤りであることも多く、これに再実行する前に利用者が気付くこともあろう。この場合は、利用者が確実にダンプファイルを消去すればいいのであるが、そうしないことも多い。そこで、通常の汎用大型計算機の出力キューに関する制御と同じように、ある一定期間保存して使用されない再実行可能なダンプファイルは自動的に消去するようにすればよい。このような作業は記憶装置のアクセス時間に関係なく、またＣＰＵを介在することはないので計算機全体の効率を落とすことはない。

なお、階層的な記憶媒体の構成は利用者の恒久的なファイルを保存する装置として、既に装備している計算機システムが存在する。この場合、ハードウェア的にはこれをそのまま使用することができる。但し、利用者の要求でファイルにアクセスにいく通常の利用法と計算機資源が枯渇したためにＯＳ側の都合でアクセスにいく本発明の利用法とがあるので、特にＣＰＵに直結される半導体記憶装置の場合はマルチポートアクセス可能である必要がある。さもないとアクセス権の調停を行う必要があるし、アクセス権が取れるまでＣＰＵがなにもしないで待つ事態も生じるからである。

以上の電子計算機システムにおける基本的な動作を、第３図のフローチャートを参照して簡単に説明しておく。

まず、通常動作において、利用者が予め設定した計算機資源の使用量が規定値に達したか否かを判定する。計算機資源の使用量が規定値に達したら、前述した中断前処理を行い、記憶装置１（本実施例では半導体メモリ４１）にロードモジュール，全レジスタ及びステータスワードレジスタの内容をフルダンプする。これと同時に処理を一時的に終了させると共に、計算機資源が足りなかった旨を利用者に通報する。この時点から、一定時間内に利用者から再実行の指令がなされたら、即ち記憶装置１にストアされた期間が規定値に達するまでに再実行の指令がなされたら、前述した再起動前処理を行い、再起動を行って通常動作に戻る。

上記一定時間内に再実行の指令がなされなかったら、記憶装置１の内容を記憶装置２（本実施例では磁気ディスク４２）にマイグレートさせ、次々にアクセス時間のより長い記憶媒体に該ダンプ内容をマイグレートさせる。個々の記憶装置２～Ｎでは、記憶装置１と同様に該記憶装置にストアされた期間が規定値に達するまでに再実行の指令がなされたら、再起動前処理，再起動を行い通常動作に戻る。そして、最終の記憶装置Ｎ（本実施例では磁気テープ４３）にマイグレートさせた後に一定時間内に再実行の

指令がなかったら、予め決められた手順に従ってファイルを消去する。

また、以上の説明では、ＣＰＵ時間を計算機資源の例として記述したが、その他の計算機資源でも同様である。記憶容量が計算機資源である場合について、簡単に述べておく。

記憶容量が計算機資源である場合は、二通りのケースが考えられる。第一は主記憶領域であり、第二は外部記憶装置である。前者の場合は今の計算機は一部のもの（特にスーパーコンピュータ）を別として、仮想記憶方式になっており、物理メモリとは基本的に無関係になっている。この仮想記憶方式でも対応し切れない主記憶領域を要求するジョブは基本的に対応し切れず、メモリ空間のアドレッシング法を抜本的に改良するほかはなく、これはアーキテクチャの全面変更を意味するので、ここでは考慮しない。

一方、後者の場合は、特に順番探査されるファイルでは、まず書き出し要求があればそれに応じてどんどん書いていく。そして当初の計算機資源を使い切った段階で、ファイル書き出しの正常終了を意味する制御データ（エンドオブファイル：ＥＯＦ）を書き込み、外部記憶装置への書き込みを一旦停止する。この段階で計算機資源は使い切っているのであるから、先に述べたように、利用者定義のプログラムと使用しているレジスタ類及びＰＳＷなどの内容のダンプを取り、これを再起動に備えて記憶装置に掃き出す。書き込んでいた記憶媒体に最後にＥＯＦマークを書き込むのは、このようにしないと再起動の後に書き込んでいたデータの一番最後を認識することができなくなることを防ぐためである。

再起動が要求されたときはプログラムのロードとその中断点に応じたレジスタ内のデータの再ロードなどをまず行う。次に、書き出し要求で中断していたのであるから、プログラムカウンタはFORTRAN でいえばWRITE 文の実行を示している筈であるが、同じファイルに追加していく場合は、まずプログラムカウンタの位置に無関係に開いているべきファイルを対応する外部記憶装置上でサーチしてその場所を探し、次にＥＯＦまで空読みしてポンインターを移動する。次に、一レコードだけバックスペースして最後に書かれたＥＯＦレコードに上書きできるようにポインターの位置を決める。ここまでは中断点からの再実行を行うための前処理である。後は通常動作に戻ってプログラムカウンタの指示通りに実行していけばよい。

これを行うには、ジョブが中断した段階で、利用者が要求したファイルが容量不足であったことを理解して、パーマネントファイルの大きさを適当に大きくしておくことが今の実施例では前提となっている。この作業用をシステム側に行わせる方法もある。例えば、再起動が要求されたら、計算機は先程まで書き込んでいたファイルとはべつの一時的なファイルに先程の中断点から書き込む。ここでまたファイルがパンクしたらその旨メッセージを出す必要があるが、ここでは正常に終了したとしよう。この結果、本来一つであるべきファイルが二つに分けられて作成された訳であるから、最後にこれまで作ったファイルを連結する（コンカティネートする）作業が必要である。ファイル連結は通常のＯＳで供給されるごく普通のユーティリティーであるから、ファイルの名前さえ保存しておけばこれは簡単に実現できる。このようにすれば利用者は再起動を要求するだけでよい。しかしこの場合は、ファイルの連結という後処理が必要なわけである。

再実行に備えてフルダンプされたファイルは初めに高速アクセス可能な記憶媒体上に置かれるが、利用者がある一定時間の内に指示を出さないと、よりアクセス速度の遅い記憶媒体に自動的にマイグレートさせられていき、最終的な記憶媒体上でも一定期間待ってみて、指示が無ければファイルが消去されてしまうことはＣＰＵ時間が計算機資源となっている場合と同じである。

このように本実施例によれば、予め設定した

計算機資源を使い切った場合でも、その中断点からの再実行が容易に行え、これまでは計算機資源の使い果たしによる異常終了（アボート）で終り、もう一度頭から計算し直さなくてはならなかったものが、それまでの計算結果を生かした形で計算を続行することができる。さらに、計算機資源を実質的に制限しないにも拘らず、細切れ作業が容易となるので、不必要な巨大ジョブの投入を減らす効果もある。一方、利用者から見れば、再計算に掛かっていた時間を大幅に短縮することができると同時に、デバック作業の効率化にも役立てることが可能であり、従って計算機の利用効率が大幅に向上することになる。

なお、本発明は上述した実施例に限定されるものではない。計算機資源はＣＰＵ時間や外部記憶装置に限るものではなく、これ以外のものを用いることもできる。実施例でも述べたように計算機資源によっては中断点から再実行させるためには適当な前処理や後処理が必要なこともある。この手順の詳細は対象とする計算機資源の種類に依存するので一概に手続きを規定することはできないが、計算機資源が枯渇した時に、中断点からの再実行を目的とし、これに備えた動作を行うＯＳであればそれは本発明の変形例であることはいうまでもない。また、計算機のシステム構成例は第１図に何等限定されるものではなく、仕様に応じて適宜変更可能である。その他、本発明の要旨を逸脱しない範囲で、種々変形して実施することができる。

［発明の効果］

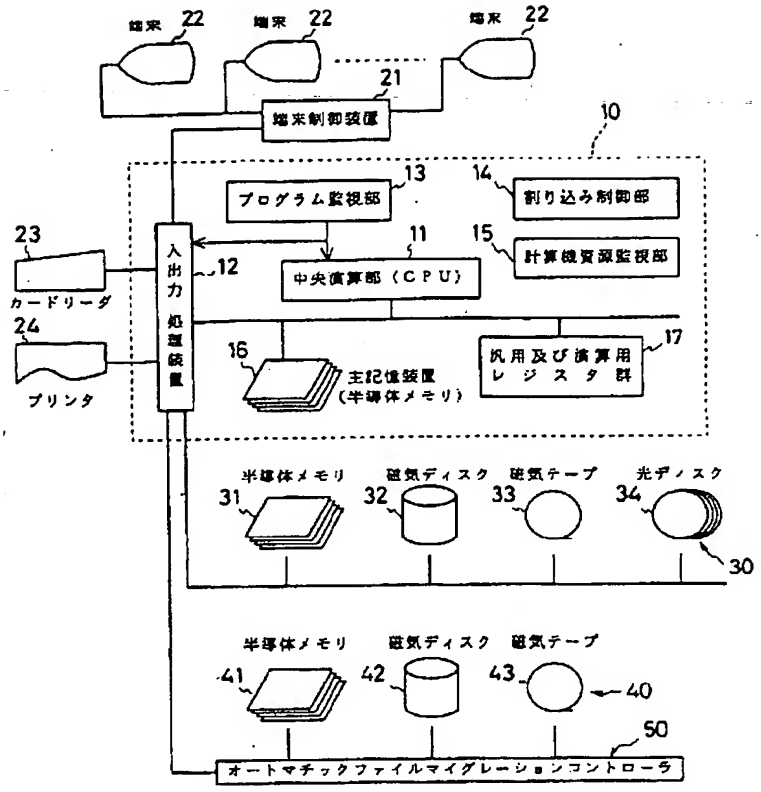
以上詳述したように本発明によれば、計算機資源を使い切ったときにそのジョブを異常終了させて実行を打ち切るのではなく、その時点で、利用者側の判断を仰ぐことを取り入れたオペレーティングシステムを構築しているので、予め設定した計算機資源を使い切った後でも、その中断点からの再実行を可能とすることができ、再計算などの無駄を省き計算機の利用効率の向上をはかることが可能となる。

4．図面の簡単な説明

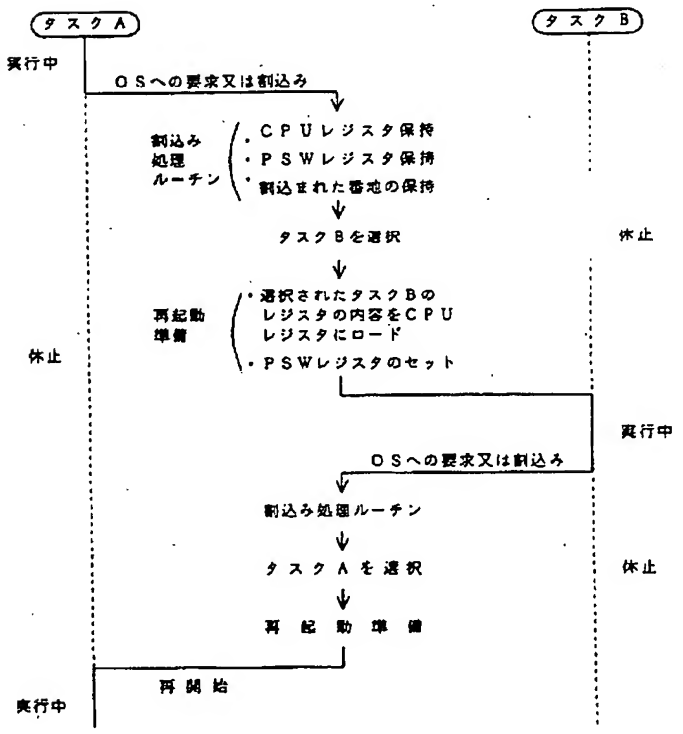
第１図は本発明の一実施例に係わるオペレーティングシステムを使用した電子計算機のシステム構成例を示す図、第２図はタクス切替えの概念を示す図、第３図は上記実施例における基本的な動作を説明するためのフローチャートである。

１０…計算機本体、
１１…中央演算部（ＣＰＵ）、
１２…入出力処理装置、
１３…プログラム監視部、
１４…割り込み制御部、
１５…計算機資源監視部、
１６…主記憶装置、
１７…汎用及び演算用レジスタ群、
２１…端末制御装置、
３０，４０…外部記憶装置、
３１，４１…半導体メモリ、
３２，４２…磁気ディスク、
３３，４３…磁気テープ、
３４…光ディスク、
５０…オートマチックファイルマイグレーションコントローラ。

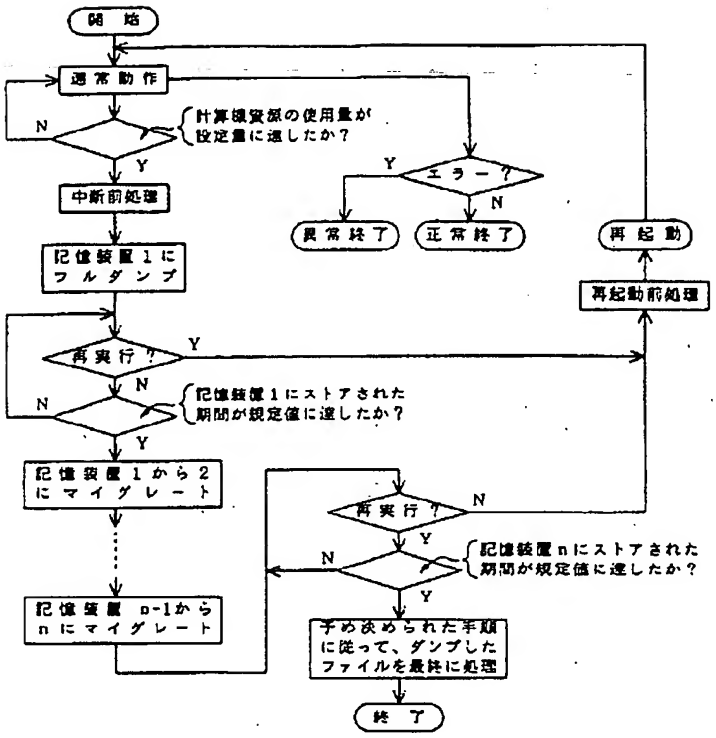
出願人代理人　弁理士　鈴江武彦


端末 22
端末 22
端末 22
21
端末制御装置
10
プログラム監視部 13
割り込み制御部 14
11
中央演算部（ＣＰＵ） 15
計算機資源監視部
23
カードリーダ
入出力処理装置 12
24
プリンタ
16
主記憶装置（半導体メモリ）
汎用及び演算用レジスタ群 17
半導体メモリ 磁気ディスク 磁気テープ 光ディスク
31 32 33 34
30
半導体メモリ 磁気ディスク 磁気テープ
41 42 43 40
50
オートマチックファイルマイグレーションコントローラ

第 1 図


タスクA
タスクB
実行中
ＯＳへの要求又は割込み
割込み処理ルーチン
・ＣＰＵレジスタ保持
・ＰＳＷレジスタ保持
・割込まれた番地の保持
タスクＢを選択
休止
再起動準備
・選択されたタスクＢのレジスタの内容をＣＰＵレジスタにロード
・ＰＳＷレジスタのセット
休止
実行中
ＯＳへの要求又は割込み
割込み処理ルーチン
タスクＡを選択
休止
再起動準備
再開始
実行中

第 2 図


開始
通常動作
計算機資源の使用量が
設定量に達したか？
N
Y
中断前処理
記憶装置1に
フルダンプ
エラー？
Y
N
異常終了
正常終了
再起動
再起動前処理
再実行？
Y
N
記憶装置1にストアされた
期間が規定値に達したか？
N
Y
記憶装置1から2
にマイグレート
記憶装置 n-1から
nにマイグレート
再実行？
N
Y
記憶装置nにストアされた
期間が規定値に達したか？
N
Y
予め決められた手順
に従って、ダンプした
ファイルを最終に処理
終了

第 3 図